propose a multi-task video understanding framework (video classification and video caption) and also discusses the influence of video frames for different methods

Video Understanding: From Video Classification to Captioning

Jiajun Sun
Stanford University
jiajuns@stanford.edu

Jing Wang
Stanford University
jingw2@stanford.edu

Ting-chun Yeh
Stanford University
chun618@stanford.edu

Abstract

In recent years, Internet users spend an amount of time on videos. However, too many videos are quite difficult for human beings to categorize and caption. Additionally, humans need to quickly recognize what kinds of videos and decide if they are going to watch them, which then require advanced techniques to do video classification and captioning. In this paper, we have explored methods and architectures to understand videos. The automatically categorization and caption are helpful for users to have better experiences when they are watching videos. The purpose of this paper is to understand how to categorize and caption the video automatically. We also propose the models to do both classification and caption in one architecture.

1. Introduction

Watching videos is popular among Internet users and their needs have generated a tremendous amount of data. Thus, advanced deep video understanding application is encouraged by the availability of a large number of annotated videos. In [2], Zuxuan Wu *et al*. review two significant research areas on the comprehension of videos: video classification and video captioning. Video classification, as introduced in this article, focuses on automatically labeling videos based on video contents and frames, while video captioning is to generate short descriptions for videos and capture dynamic information such as human actions and car trajectories. Specifically, a big challenge in video classification and captioning is to fuse different video frames over time. In video classification, researchers come up with methods in three dimensions: temporal feature pooling (TFP) [3] and 3D convolution network (C3D) [6], while sequence to sequence model are applied in video captioning [7].

Although several approaches are available, few researchers attempt to study the number of frames in video classification and build up an architecture to do multi-task. The purpose of this paper is to build up an architecture to do multiple tasks: video classification and captioning. We also give intuitions about the effects of the number of frames on classification accuracy. In this project, we implement frame majority vote, temporal feature pooling (TFP) [3], 3D Convolution (C3D) [6], and Long Short Term Memory (LSTM) [9] to classify videos. We further conduct sequence to sequence model in video captioning. The experiments are run on the Microsoft multimedia challenge dataset. This dataset provides short-clip (around 10-15 seconds) videos rather than long videos. The output in video classification is the predicted video categories, and the output in video captioning is the predicted word index in the trained vocabulary and then video descriptions. Finally, we compare different methods and factors and analyze the effects of the corresponding factors on model performance.

2. Related Work

Unlike image classification, video classification has sequential frames input. The basic idea is to classify every frames per video and then output the video category based on the results of majority vote [1]. In this case, video classification is treated as image classification, which, however, cannot keep the sequential information of videos. Therefore, in our project, frames majority vote method is the baseline method. In terms of fusing sequential frames, Yue-Hei Ng *et al*. clearly describe two approaches: 3D temporal max pooling and Long Short Term Memory (LSTM) [3]. It turns out LSTM [3] can maintain the dynamic contents of videos to make more accurate predictions. The authors also couple the optical flow into LSTM model to obtain higher classification accuracy. Feichtenhofer *et al*. come up with 3D Convolution + 3D Max Pooling method to fuse frames, which is able to combine spatial and temporal information simultaneously [4]. Another deep learning approach, which is called two stream CNN, proposed by Simonyan *et al*. [13], has two streams of CNN. One contains spatial information using single frame and the other one contains temporal information using optical flow . However, we found optical flow are useful in classifying different actions and our video are short clips, most of which does not contain specific and continuous actions. Therefore, those model using optical flow does not applicable to our project.

3D Convolutional Neural Network (CNN) also operates on stacked video frames. It extends the original 2D convolutional kernel and 2D pool kernel into 3D kernel to capture both spatial and temporal space. C3D model proposed by Du Tran *et al*. [6], which achieves state-of-the-art performance on video classification problem. However, training a 3D CNN is time consuming and the spatial-temporal structure in videos may be too complex to capture and it gives only a moderate improvement compared to single frame method [5].

In video captioning, applying RNN to translate visual sequence to natural language is largely motivated by the work in Neural Machine Translation [14]. The idea is to treat visual inputs from CNN such as VGG [12] as source texts and captions as target language output. Venugopalan *et al*. [7] average the sequential outputs from pretrained CNN for every frames per video, which is called mean pooling layer. And then, he inputs the average results into two stacked LSTM to generate the descriptions. Sequence to sequence model, which is introduced from machine translation [14, 15], is also applied by Venugopalan *et al*. [8] on Microsoft Video Description corpus (MSVD), the MPII Movie Description Corpus (MPII-MD) [11], and the Montreal Video Annotation Dataset (M-VAD). The results, as shown in this article, outperform the visual labels and mean pooling method. Some other methods have been proposed to solve video captioning problem such as hierarchical RNN [17, 18]. However, this method cannot easily share architecture with video classification task. Therefore, in our project, we accept sequence to sequence model to do video captioning.

3. Approach

3.1. Video Classification

Frames Majority Vote

Video classification can be simply treated as image classification. Every frame per video can be labeled as the same video category. And then we train CNN model using frames datasets. In test mode, every frame per video is predicted as a video category so that the video type can be obtained from the majority vote of frame predictions (see equation 1. Specifically, assume we have 15 frames for a certain video. 10 frames are classified as *news*, and 5 are *action movies*. By majority vote, the video is classified as *news*. Our frame majority vote model architecture is shown in Table 2.

$$C = M_{i=1}^{f} c_i \tag{1}$$

where C is the category of the video, M represents majority vote, f is the frame number, c_i is the video category of the frame i.

Temporal Feature Pooling (TFP)

Temporal Features Pooling was introduced from bag-of-words representation application [3], which is a layer to concatenate outputs from Convolutional Neural Network (CNN) with video frames input.

Yue-Hei Ng *et al*. propose five temporal feature pooling architecture: Convolution Pooling, Late Pooling, Slow Pooling, Local Pooling and Time-Domain Pooling [3]. Convolution Pooling (Figure 1) with max pooling performs best after running experiments in UCF-101 and Sports-1M datasets.

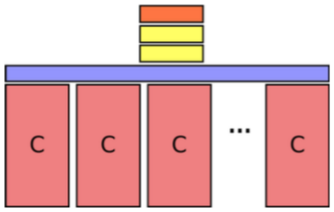

Figure 1: Temporal Pooling Layer [3] (Red: CNN; Blue: Temporal max pooling; Yellow: Fully-connected; Orange: Softmax)

In details, CNN architecture gives outputs for every frame of a video. Assume we have $x \in \mathbb{R}^{H \times W \times T \times D}$, which is attained by concatenating outputs from pretrained CNN model. Then, we apply max-pooling to x within a 3D pooling cube $H' \times W' \times T'$, which is an extension of 2D pooling to the temporal dimension. Note that no pooling is used across the channel dimension [4] . In this project, we use VGG16 as our pretrained CNN architecture and apply $3\times$ 2 $\times 2$ temporal max pooling with stride 3, 2, and 2, respectively. The corresponding dimensions of data are Time (T), Height (H), and Width (W). The entire architecture can be seen in Table 3.

3D Convolution (C3D)

A simple and effective approach proposed by Du Tran *et al*. [6] is to use deep 3-dimensional convolution neural network trained on a large scale video dataset. C3D operates on stacked video frames and extends the original 2D convolution kernel and 2D pooling kernel into 3D kernel to capture both spatial and temporal information. C3D achieves state-of-the-art on video classification problems. However, training a 3D CNN is very time consuming and the spatial-temporal structure in videos may be too complex to capture.

According to Du Tran *et al*., the structure with $3 \times 3 \times 3$ convolution kernels in all layers gives the best performance. We use the suggested $3 \times 3 \times 3$ convolution kernel size. However, different from original structure, we add batch normalization layers into C3D and expect to get more

stable result.

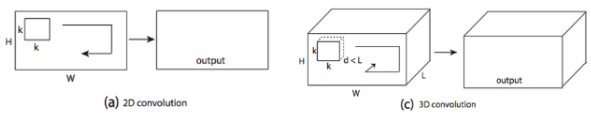

Figure 2: 3D convolution

Figure 2: C3D model (convolution on temporal and spatial data)

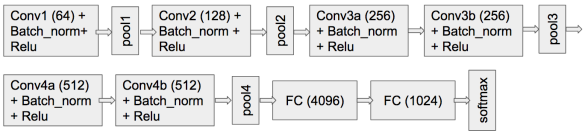

Figure 3: C3D model architecture (only pool1 is $1 \times 2 \times 2$, all other pool kernels are $2 \times 2 \times 2$. All Conv are $3 \times 3 \times 3$)

LSTM

Another popular approach for video classification is to use LSTM. Similar to temporal feature pooling, LSTM networks operate on frame-level CNN activations as well as integrate information over time [3]. LSTM also outputs a hidden vector for each input activation frame. As shown in Figure 4, each cell (c) in LSTM layers accepts stacked h_{t-1} and x_t as inputs. The inputs enter four gates after dot producting with weights (W). Each gate has different functions[1].

Mathematically, LSTM cell can be expressed as the following equations [9]:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

where σ represents sigmoid function, and $\odot$ is element-wise multiplication. Compared to the vanilla recurrent neural network, LSTM has uninterrupted gradient flow, which is easier to back-propagate. LSTM is also more stable without gradient exploding or vanishing [2]. Referring to [3], we build up our LSTM architecture for video classification, as shown in Figure 4 and Table 4. The first LSTM layer accept the inputs from the pretrained CNN model, and then the second LSTM receives sequential outputs from the previous LSTM layer. In the end, we apply the output into fully-connected layer and calculate the final softmax score.

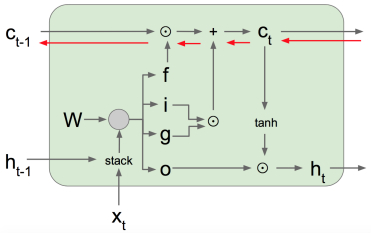

Figure 4: LSTM cell [9] (c: cell, f: forget gate, i: input gate, g: 'gate' gate, h: hidden layer, o: output gate, W: weights)

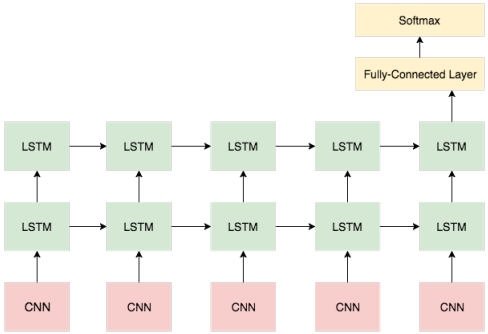

Figure 5: LSTM architecture sketch [3] (LSTM layers (green) takes the output from the final CNN layer (pink) at each consecutive video frame. CNN outputs are processed forward through time and upwards through stacked LSTMs. A softmax layer (yellow) predicts the class at each time step.)

3.2. Video Captioning

Sequence to sequence model is introduced from machine translation, which is used to generate captions for videos here. We stack two LSTM layers to learn representation of a sequence of frames. One is to encode the inputs from the pretrained CNN model. The other is to decode the outputs from encoder into a sequence that generates descriptions for videos. [7].

In the first several time steps, the top layer LSTM (marked blue in figure 6) receives frame input from VGG16 and encode them into LSTM cell state and output. At the same time, the bottom layer LSTM (marked in orange) takes in a vector (which is concatenated from word padding and LSTM outputs). There is no loss during the encoding stage. At the decoding stage, top LSTM cell receive

[1] CS231n Lecture 10 Recurrent Neural Network
[2] CS231n Lecture 10 Recurrent Neural Network

padding frames while bottom LSTM start to generate predicted words. Softmax with cross entropy is used as our loss function.

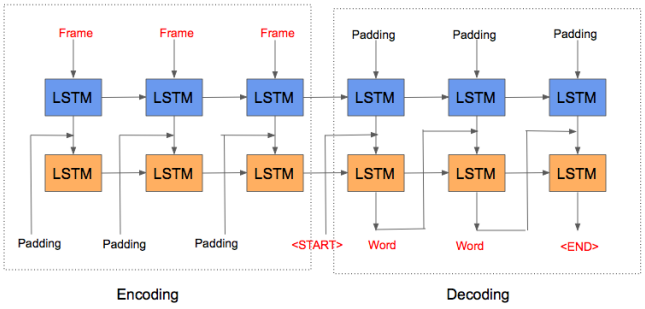

Figure 6: Sequence to sequence model, with VGG16 frame feature vector as input and predicted word as output

3.3. Multi-task Architecture

In the video captioning task, our sequence to sequence model is structured to be capable to handle different deep learning tasks[1]. As shown in figure 7, the encoder part of the model can be used for video classification. This model can be trained for video captioning if both encoder and decoder are used. When the input has only 1 frame, this model can be treated as image captioning model. Since all the models are sharing the same code base, they can also share weights. This will be mentioned in training process section, we trained video captioning model first using COCO image captioning data and used COCO image captioning weights as our initialization for video captioning model.

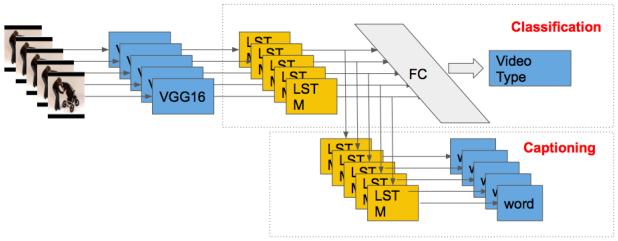

Figure 7: Muti-task model architecture

[1]The codebase has referred to the General NLP Model Class in this github repo: https://github.com/jiajuns/cs224n-project

4. Experiment

4.1. Datasets

The video dataset is from Microsoft Multimedia Challenge [3], which is a Large Video Description Dataset for Bridging Video and Language. This dataset has 10,000 video clips from 20 different categories with a total duration of 41.2 hours. Annotated description has in total of 200,000 sentences. Labels for this dataset, including clip category and annotated description, are produced by Amazon Mechanical Turk (AMT) workers. Each clip has 20 natural sentences by 1327 AMT workers.

However, not all videos are available at this time. Only available videos are downloaded and processed in our project. Additionally, as you can see the distribution of video categories (Figure 8), we have extremely unbalanced data. To address this, the video categories with low counts are removed first in order to obtain more balanced data. After that, we re-mapped previous categories indice to the new indice for model training. In the end, 6100 videos are in processed data, of which 4270 are to train and validate and the rest are to test. The final categories we kept: music, gamming, sports/actions, news/events/politics, movies/comedy, vehicles/autos, howto, animals/pets, kids/family, and food/drink. The example of data is shown in 1. We also resized every frame per video to $224 \times 224 \times 3$ in order to fit into pretrained VGG16 model.

In terms of word embedding vectors, we applied pretrained GloVe Wikipedia 2014 + Gigaword 5 [4] [16] and filter the words outside our video descriptions.

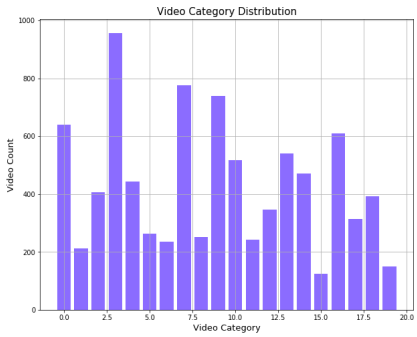

Figure 8: Distribution of Video Categories (X-axis: video categories indice. Video categories distribute unevenly, thus data should be processed first)

4.2. Video Classification Training Process

Before training video classification model (including temporal pooling and LSTM model), we sample 10 or 15

[3]http://ms-multimedia-challenge.com/2017/dataset
[4]https://nlp.stanford.edu/projects/glove/

Table 1: Video Data Example

Items	Contents
category	9
url	https://www.youtube.com/ watch?v=9lZi22qLlEo
video_id	video0
start time	137.72
end time	149/44
id	0
caption	a man drives a vehicle through the countryside
sen_id	109462

frames out of the video clips. Then a VGG16 ConvNet runs through each frame and we take the last ConvNet output as a vector representation of a frame. Those vector representation is treated as the classification model input and use video category as our predicted labels. When we are training each model, the structure of Temporal pooling is listed in Table 3 and the structure of LSTM is listed in Table 4. For C3D model, the structure was mentioned in Figure 3. We found it was quite hard to train a good C3D model, which required enough data for training and take much time. Unfortunately, we don't have enough data and time to train a good C3D model. In order to make C3D training work, we shrink the images to smaller size that our GPU can cope with. When we visualize images, shrinking makes images blur so that our C3D model has low accuracy.

Table 2: Frame majority vote model architecture

Layer	Output
VGG16 without last 1000 unit layer	1×4096
Fully-connected layer	$1 \times$ number of classes
Softmax	Scores

Table 3: CNN-based temporal max pooling classification architecture

Layer	Output
VGG16 without 3 fully-connected	frame number $\times 7 \times 7 \times 512$
Temporal max pooling (padding = 'VALID')	$3 \times 3 \times 3 \times 512$
Fully-connected layer	1×4096
Fully-connected layer	$1 \times$ number of classes
Softmax	Scores

Table 4: LSTM classification hyperparameters (Those number should be modified)

Layer	Output
VGG16 without fully-connected layers	frame number $\times 7 \times 7 \times 512$
LSTM	128
LSTM	128
Fully-connected layer	$1 \times$ number of classes
Softmax	Scores

4.3. Video Captioning Training Process

First, we didn't check the correctness of caption from the dataset which result in terrible trained model. Then, we found out there are many misspelled words in the labeled captions and even some unrecognized Spanish words. Therefore, we decided to clean the data and correct words. After that, we reduced 25000 words to 16000 words.

Since there are some words that are not in the GloVe 6B embedding, we tried to train word embedding matrix from the ground up. However, the training results are not satisfactory, we decided to convert these words to $< unk >$ in GloVe 6B embedding and stop training embedding matrix.

During our experiments, we found it is extremely hard to train video caption model end-to-end since our video-caption dataset is not large enough. Therefore, we decided to train sequence to sequence model in two stages. At the first stage, we used Microsoft COCO data and GloVe 6B embedding to train the captioning model. Since our architecture is capable of handling both video and image captioning task, weights of both model can be shared between each other. When finish training our model on COCO dataset, we take the variable weights as the initialization of video captioning model.

4.4. Evaluation Metrics & Results

4.4.1 Classification Accuracy

The basic metric for classification problems is to measure the accuracy, which is to calculate the percent of correct predictions. In this project, we adopt this accuracy as the basic evaluation metric. We cited part of codes to plot accuracy and loss curves [5].

4.4.2 Precision-Recall

Precision-recall is a common method in classification problems. Precision measures result relevancy, while recall measures how many relevant results are returned [6]. F1-score is

[5] http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/

[6] http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

weighted harmonic mean of the precision and recall, and it also measures the degree of balance between false positives and false negatives. For a certain class, we have the following table (see Table 5) and equations. then

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

where TP is true positive, FP is false positive and FN is false negative.

Table 5: Confusion Matrix

	Positive	Negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

4.4.3 BLUE

BLUE is a algorithm for evaluating quality of a text generated by machine. Scores are calculated for each generated segments by comparing with reference text [19]. BLUE score for a candidate sentence is expressed as:

$$p = \frac{m}{w_t} \tag{5}$$

Where m is the number of words in the candidate sentence that are also found in reference; and w_t is the total number of words in the candidate sentence.

In practice, rather than counting the occurrence of each individual word, n-grams counts could be more useful. When calculating n-grams BLUE score, just need to count the occurrence of n-grams.

4.4.4 Video Classification

We applied several different approaches in video classification. The results are shown in Table 6 and Figure 9, 10 and 11.

4.4.5 Image Captioning

We showed our COCO image caption results below and compared it with the result from the state-of-art model [10].

4.4.6 Video Captioning

We showed our video captioning results below and calculate the BLEU scores which are listed in Table 8

Table 6: Video classification results (%)

Methods	Items	10 frames	15 frames
VGG16 + TFP	Accuracy	52.6	57.0
	F1-score	51.0	57.0
VGG16 + LSTM	Accuracy	59.6	61.5
	F1-score	60.0	61.0
VGG16 frames (majority vote)	Accuracy	57.8	
	F1-score	57.0	
C3D	Accuracy	38.8	-

	precision	recall	f1-score	support
music	0.33	0.37	0.35	195
gaming	0.52	0.62	0.57	122
sports/actions	0.69	0.71	0.70	288
news/events/politics	0.45	0.49	0.47	133
movie/comedy	0.51	0.58	0.54	220
vehicles/autos	0.73	0.77	0.75	229
howto	0.51	0.45	0.48	163
animals/pets	0.62	0.50	0.55	145
kids/family	0.51	0.38	0.44	152
food/drink	0.72	0.64	0.68	183
avg / total	0.57	0.57	0.57	1830

Figure 9: Classification report in TFP with 15 frames

	precision	recall	f1-score	support
music	0.41	0.53	0.46	195
gaming	0.59	0.67	0.63	122
sports/actions	0.74	0.72	0.73	288
news/events/politics	0.59	0.47	0.53	133
movie/comedy	0.49	0.59	0.54	220
vehicles/autos	0.73	0.82	0.77	229
howto	0.69	0.44	0.53	163
animals/pets	0.62	0.50	0.55	145
kids/family	0.54	0.45	0.49	152
food/drink	0.69	0.68	0.68	183
avg / total	0.62	0.61	0.61	1830

Figure 10: Classification report in LSTM with 15 frames

	precision	recall	f1-score	support
music	0.53	0.29	0.37	195
gaming	0.52	0.64	0.58	122
sports/actions	0.62	0.77	0.69	288
news/events/politics	0.45	0.45	0.45	133
movie/comedy	0.43	0.62	0.51	220
vehicles/autos	0.77	0.77	0.77	229
howto	0.64	0.48	0.55	163
animals/pets	0.58	0.50	0.53	145
kids/family	0.49	0.51	0.50	152
food/drink	0.79	0.55	0.65	183
avg / total	0.59	0.58	0.57	1830

Figure 11: Classification report in frame majority vote

4.5. Discussion

4.5.1 Video Classification

10 and 15 frames are extracted as the input for TFP and LSTM model. From the video classification result (Table 6 and Figure 9, 10 and 11), LSTM performs better than TFP because it can maintain temporal contents of video in its

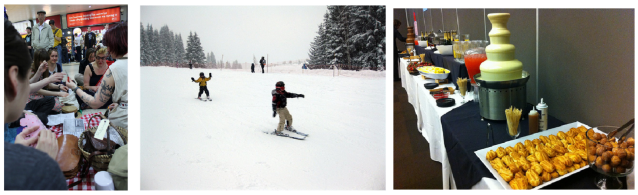

Figure 12: Predicted caption: (left)a group of people siting around a table; (mid)a couple of people on on a snow; (right)a plate table with down holding of food on it

Table 7: Image caption results (MS COCO c5, %)

Score function	Our Results	State-of-art[10]
BLEU-1	62.8	75
BLEU-2	42.6	58
BLEU-3	29.4	43
BLEU-4	15.7	32

Figure 13: Predicted caption: a person is folding paper

Figure 14: Predicted caption: two men are wrestling

Figure 15: Predicted caption: a man playing a basketball

Table 8: Video Caption Results (%)

Items	BLEU-1	BLEU-2	BLEU-3	BLEU-4
score	65.2	57.6	48.2	39.4

cell states. TFP is able to achieve bigger improvement in test accuracy (4.4%) than LSTM (1%) when the number of frames increase, which is because TFP directly relates to the frame number. TFP is more sensitive to the number of frames than LSTM. Our baseline approach is using majority vote of 15 frame. This basic method achieves quite good results, which beats 10 frames temporal pooling and C3D. It is important to note that the low accuracy of C3D model is due to our training process. Other models use pretrained VGG16 weights as initialization. However, C3D model is trained from scratch. Additionally, C3D training requires more video data than we could provide.

From figure 9, 10 and figure 11, we observed that MUSIC category has the lowest F1 score in all models we tried. The audio information may be helpful for MUSIC category.

4.5.2 Video Captioning

The video descriptions are not clean. As we said before, many misspelled words exist so as to force us to examine the data and figure out how to clean it. We also observed that some videos need audio information for prediction. However, we can only use image frames in this project.

Since our model is able to handle multi-task, we first train our model on COCO image datasets, then use the trained weights as the initialization of video caption model. In COCO c5 datasets, there are only 1004 words, while there are 15000 words in our video datasets. In order to share the weights, we make two models sharing the same word-embedding. we think this may be one of the reasons that affect our model performance in image captioning.

When examining the predictions, we found out that some words appears frequently, such as "a", "a man is", "a woman is", ... and so on (see fig 16, fig 17, fig 18). We checked the captions of our dataset and found these words appear more frequently than others, which means the model tends to learn these words first and has biases on frequent pattern or words. It is also noted that when the video frames is hard to interpret (even for human), our model will output many "a"s as final predictions. To resolve this problem, we think we need to enlarge our datasets or include audio information. Although COCO dataset have thousands of images, it only consists of 1004 words in captions, which is not enough for our model to generalize well.

Besides, our model prefers to predict short sentence rather than longer sentence. One reason is that the longer sentence requires more learning and more data. Since our data is not enough, our model is likely to be conservative and mostly results in short sentences. Another reason is possibly that our model does not have an attention mechanism. For a longer sentence, the gradient is hard to propagate to the very beginning.

Figure 16: Predicted caption: a person is a a

Figure 17: Predicted caption: a car is driving a a a road

Figure 18: Predicted caption: a man is a a a a

5. Conclusion

5.1. Video Classification

To conclude, in video classification, having more frames is equivalent to having more data input so that they produce better models and higher accuracy. TFP is more sensitive to the number of frames than LSTM, which is because TFP is directly related to the number of frames. Different number of frames determine the range of dimensions that TFP has to pool. Compared to TFP, LSTM model slightly improves performance mainly because of more data input. However, we can not simply conclude that more datasets always lead to better performance. We have to consider many factors about the frame number that might affect the performance (i.e. the time range of frames). Frame majority vote treats video classification as image (frame) classification. Since the pretrained VGG16 exists, quite a number of frames input could make sure the method achieves satisfactory results, which even beats temporal feature pooling in 10 frames.

5.2. Video Captioning

This experiment demonstrated that our multi-task architecture has significant advantages when training complex video captioning models. This architecture allows us to train image captioning dataset at the first stage, and then train a video caption model using video captioning datasets with pretrained weights, which gives much better results than a end-to-end training on video captioning dataset as well as saves training time.

5.3. Future Work

To improve our model, we consider to add an attention layer to help our model find focused regions. Besides, we could do better in dataset cleaning since the dataset is not easily handled. Some videos are originally even hard for us to interpret. Some videos of our dataset need to provide audio information to make the model have better performance. In addition, it's actually rough method to extract one frame per second from video. It probably missed some important information. To have better performance, extracting more frames from video is likely to be helpful.

References

[1] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1725-1732).

[2] Wu, Z., Yao, T., Fu, Y., & Jiang, Y. G. (2016). *Deep Learning for Video Classification and Captioning.* arXiv preprint arXiv:1609.06782.

[3] Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015). *Beyond short snippets: Deep networks for video classification.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4694-4702).

[4] Feichtenhofer, C., Pinz, A., & Zisserman, A. (2016). *Convolutional two-stream network fusion for video action recognition.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1933-1941).

[5] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1725-1732).

[6] Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). *Learning spatiotemporal features with 3d convolutional networks.* In Proceedings of the IEEE International Conference on Computer Vision (pp. 4489- 4497).

[7] Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R., & Saenko, K. (2014). *Translating videos to natural language using deep recurrent neural networks.* arXiv preprint arXiv:1412.4729.

[8] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., & Saenko, K. (2015). *Sequence to sequence-video to text.* In Proceedings of the IEEE International Conference on Computer Vision (pp. 4534-4542).

[9] Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory. Neural computation*, 9(8), 1735-1780.

[10] Chen, M., Ding, G., Zhao, S., Chen, H., Liu, Q., & Han, J. (2017, February) *Reference Based LSTM for Image Captioning* In AAAI (pp. 3981-3987).

[11] Rohrbach, A., Rohrbach, M., Tandon, N., & Schiele, B. (2015). A dataset for movie description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3202-3212).

[12] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[13] Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In Advances in neural information processing systems (pp. 568-576).

[14] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[15] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[16] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global Vectors for Word Representation. In EMNLP (Vol. 14, pp. 1532-1543).

[17] Yu, H., Wang, J., Huang, Z., Yang, Y., & Xu, W. (2016). Video paragraph captioning using hierarchical recurrent neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4584-4593).

[18] Pan, P., Xu, Z., Yang, Y., Wu, F., & Zhuang, Y. (2016). Hierarchical recurrent neural encoder for video representation with application to captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1029-1038).

[19] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics (pp. 311-318). Association for Computational Linguistics.